

实验二 译码器和编码器的设计

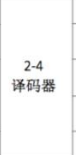
1. 实验目的

通过学习常用的译码器和编码器的设计方法以及七段数码管的使用，设计一个 8-3 优先编码器及七段数码管显示，进而促进对 verilog 和 quartus 的理解。

2. 实验原理

译码器也是组合逻辑电路的一个重要器件，译码器是将某一输入信息转换为某一特定输出的逻辑电路，通常是多路输入/输出电路，它将 n 位的输入编码转换为 m 位的编码输出，一般情况下 $n < m$ ，输入编码和输出编码之间存在着——对应的映射关系，每个输入编码产生唯一的一个不同于其他的输出编码。

例如 2-4 译码器的真值表为：

x_0		y_0	E_n	x_0	x_1	y_0	y_1	y_2	y_3
x_1		y_1	0	x	x	0	0	0	0
		y_2	1	0	0	1	0	0	0
		y_3	1	0	1	0	1	0	0
E_n			1	1	0	0	0	1	0
			1	1	1	0	0	0	1

编码器是一种与译码器功能相反的逻辑电路，编码器的输出编码比其输入编码位数少。

下图是 4-2 编码器的真值表：

x_3	x_2	x_1	x_0	y_1	y_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

七段 LED 数码管是一种常用的显示元件，常应用于手表、计算器等仪器中，用于显示数值。图 2 7 是数码管的原理图，数码管分为共阴极和共阳极两种类型，共阴极就是将七个 LED 的阴极连在一起，让其接低电平。这样给任何一个 LED 的另一端高电平，它便能点亮。而共阳极就是将七个 LED 的阳极连在一起，让其接高电平。这样，给任何一个 LED 的另一端低电平，它就能点亮。

3. 实验器材/环境

系统：windows10

开发软件：Quartus 17.1 Lite

开发板：DE10 Standard

仿真环境：ModelSim

芯片：Cyclone V , 5CSXFC6D6

4. 设计思路

首先设计一个 8-3 优先编码器，利用 for 循环，从高位到低位遍历，遇到 1 停止，记录下来位置输出，如果没有 1，指示位为 0，输出二进制结果为 000，module 代码：
(indi 为指示位，x 是输入端，y 是输出的三位二进制编码，s 为 7 段数码管结果)

```
module encoder83(x,indi,y,s);
    input [7:0]x;
    output reg indi;
    output reg [2:0]y;
    output reg [6:0]s;
    integer i;

    always @ (*)
    begin
        indi = 0;
        for(i = 7; i > 0; i = i - 1)begin
            if((x[i] == 1) && (indi == 0))begin
                y = i;
                indi = 1;
            end
        end
        if(indi == 0)begin
            y = 0;
        end
    end
end
```

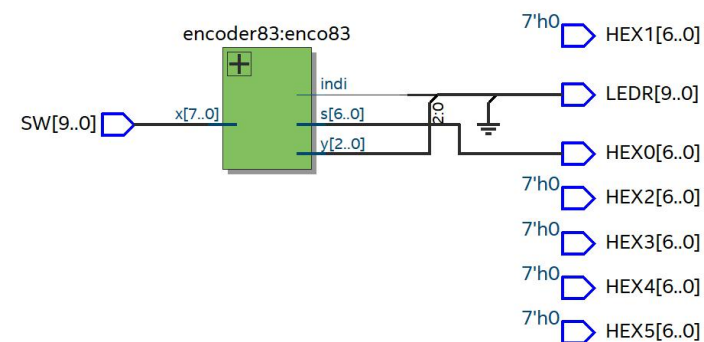
再将得到的编码结果输出到 7 段数码管上，用 case 语句，用 0~7 逐个对应赋值，代码：

```
case(y)
    0:s = 7'b1000000;
    1:s = 7'b1111001;
    2:s = 7'b0100100;
    3:s = 7'b0110000;
    4:s = 7'b0011001;
    5:s = 7'b0010010;
    6:s = 7'b0000010;
    7:s = 7'b1111000;
    default:s = 7'b1111111;
endcase
```

顶层文件调用：

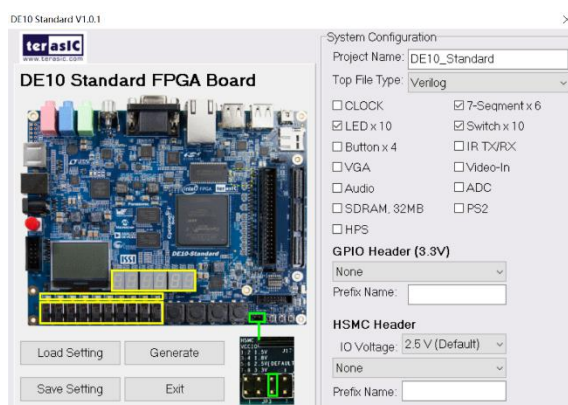
```
encoder83 enco83(SW[7:0],LEDR[3],LEDR[2:0],HEX0[6:0]);
```

RTL viewer 电路图：




5. 实验步骤

一 . 利用 systembuilder 建立工程



二. 编写 verilog 代码

在工程目录打开 qpf 文件  exp2.qpf

新建一个.v 文件, 编写 8-3 优先编码器+7 段数码管的模块 `module encoder83(x,indi,y,s);`
再将该文件名更改为 encoder83.v, (文件名与模块名一致)

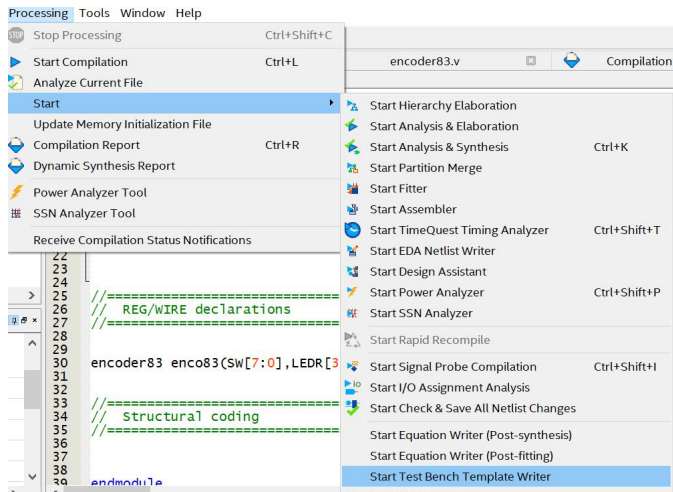
在顶层文件中根据需要调用 encoder83 模块

```
encoder83 enco83(SW[7:0],LEDR[3],LEDR[2:0],HEX0[6:0]);
```

进行编译调试, 直至成功。

三. 编写 test bench

利用 processing 中的 start test bench template writer 让系统自动生成 test bench 测试文件



在 simulation 中的 modelsim 文件中打开.vt 文件

编写.vt 文件:

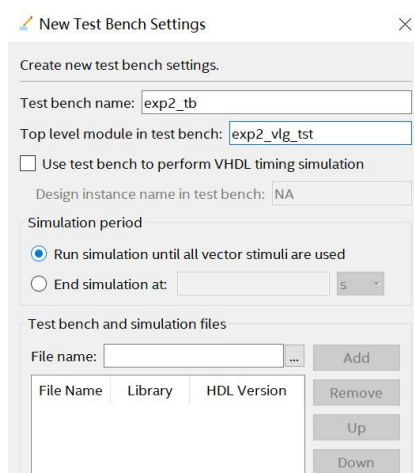
修改仿真时间单位, `timescale 10_ns/ 1_ps`

在 initial 下方 begin 和 end 之间写入测试数据

为避免风险, 将 always 语句注释。

四. 设置 test bench

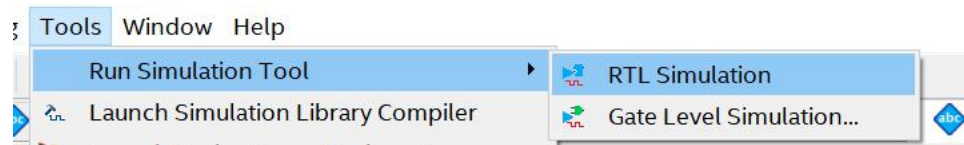
在 assignments->settings->simulation->compile test bench 中设置 test bench, new 一个 test bench, 注意将顶层实体修改为 exp2_vlg_tst



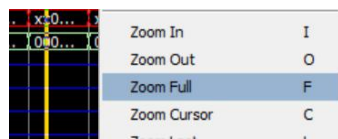
再将 simulation->modelsim 中的.vt 文件 add 进去，完成配置

五 . 仿真测试

利用 RTL Simulation 调用 modelsim



在波形图中 zoom full 就可以观察结果



六 . 利用 fpga 测试

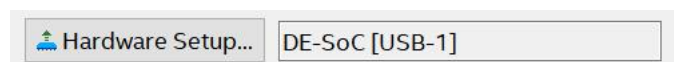
检查工程目录下有 sof 文件  exp2.sof

由于 systembuilder 已经配置好引脚，所以 sof 文件一般是自动就有了的。

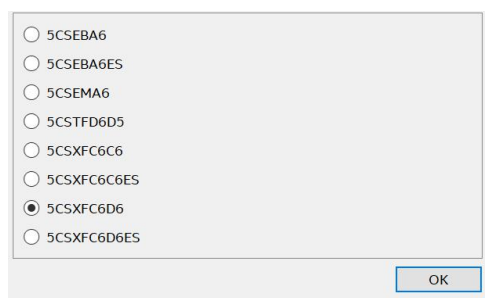
将 fpga 充电，usb 接电脑，点击 programmer 按钮(第三个)



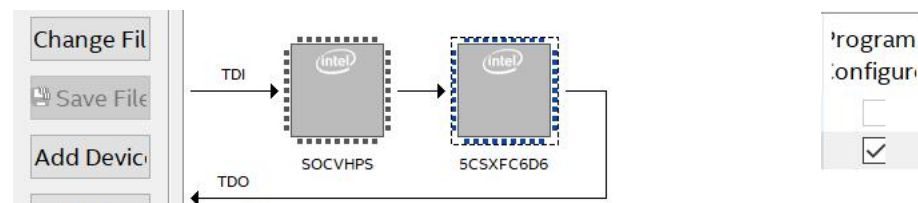
hardware 选择 usb



select device 选择倒数第二个



选中 5CSXFC6D6，change file，选中.sof 文件，接着在 program 上打勾，在点击 start 开始测试



6. 测试方法

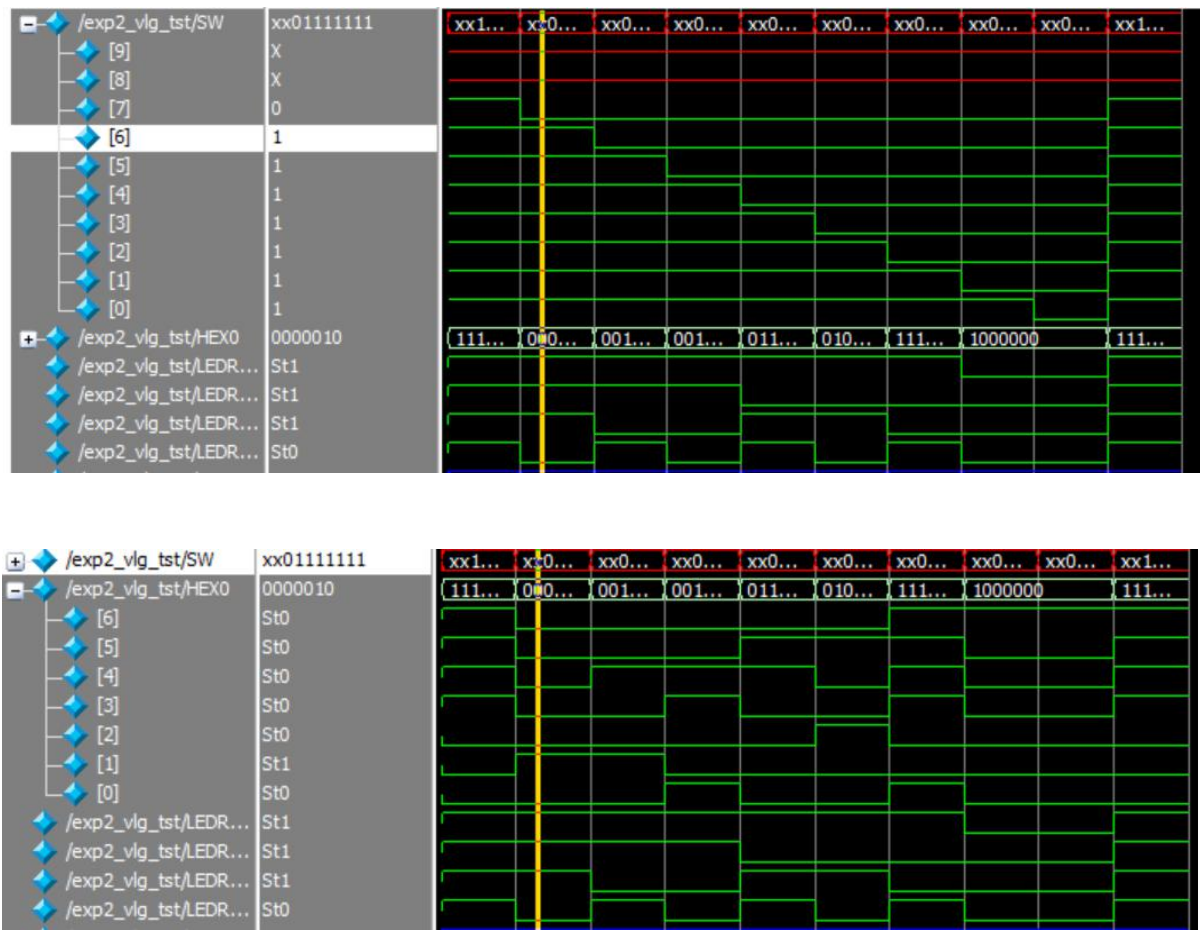
分 9 种情况，即从高位到低位数第一个 1 分别出现在 7 号~0 号 and 没有 1 的情况，这样输出

会有 8 种不同的情况，即 0~7，可以将 7 段数码管也全部测试一下。下图是测试文件代码：

```
initial
begin
// code that executes only once
// insert code here --> begin
SW[7:0] = 8'b11111111;#20;
SW[7:0] = 8'b01111111;#20;
SW[7:0] = 8'b00111111;#20;
SW[7:0] = 8'b00011111;#20;
SW[7:0] = 8'b00001111;#20;
SW[7:0] = 8'b00000111;#20;
SW[7:0] = 8'b00000011;#20;
SW[7:0] = 8'b00000001;#20;
SW[7:0] = 8'b00000000;#20;
SW[7:0] = 8'b11111111;#20;
// --> end
$display("Running testbench");
end
```

7. 实验结果

modelsim 仿真结果：

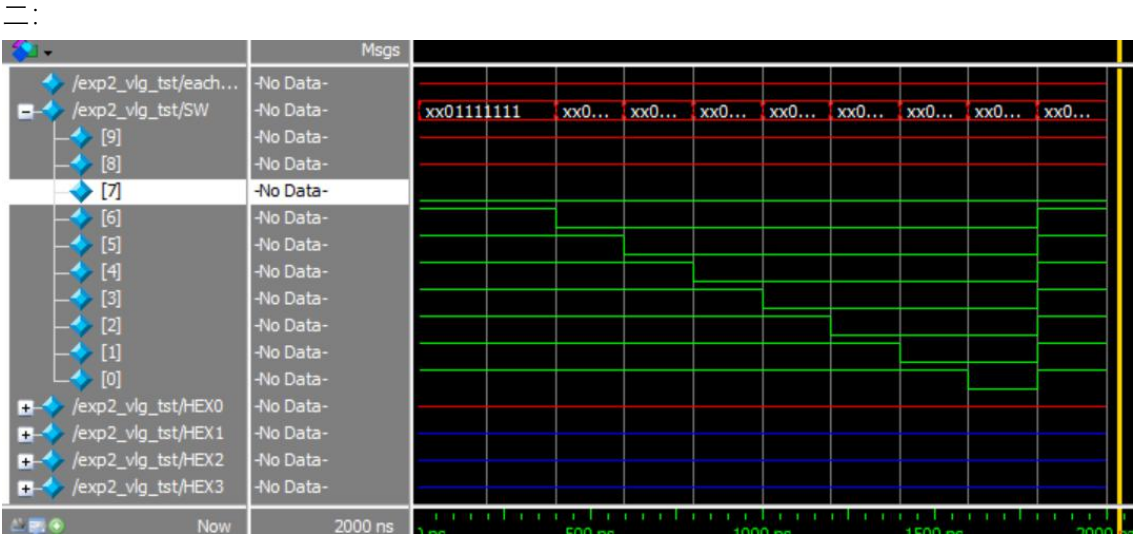


开发板运行结果：

8. 实验中遇到的问题



是由于上一个 modelsim 工程没有有关，又开了新的



本来应该是输出位的 HEX0 和 LEDR[3:0]全是 x，即无效输入，因为我在 always @ 后面的括号里写入了我要输出的量，即 always @ (indi or y or s)，括号中应当写对于输出有影响的量，是 always @ (x[7:0]) 或者直接 always @ (*)

三：开发板上七段数码管显示乱码，不是赋值语句的错误，而是 for 循环的问题

一开始是这样写的：

```

for(i = 7; i >= 0; i = i - 1)begin
    if((x[i] == 1) && (indi == 0))begin
        y = i;
        indi = 1;
    end
end

```

就会出现 $x[-1]$ 的情况，导致乱码，所以改为 `for(i = 7; i > 0; i = i - 1)`

四：一些 bug，比如没有声明 for 循环中的 i ，需要在外围 integer i 进行声明
for 循环中不支持 $i--$ 的写法，需要写为 $i = i - 1$

9. 实验得到的启示

对于 verilog 中 for 语句和 always 语句有了更深的理解
在上板测试之前一定要通过仿真测试，不然一定会失败的
可以用 case 语句来完成优先编码器的设计

10. 意见和建议

